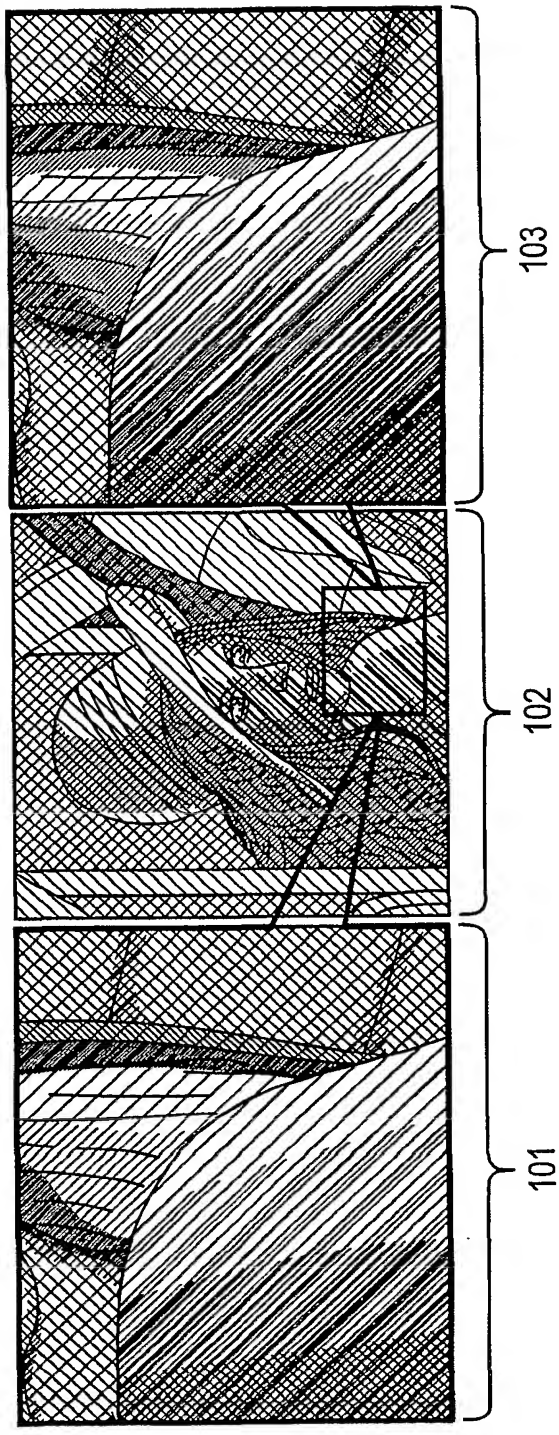

FIG. 1

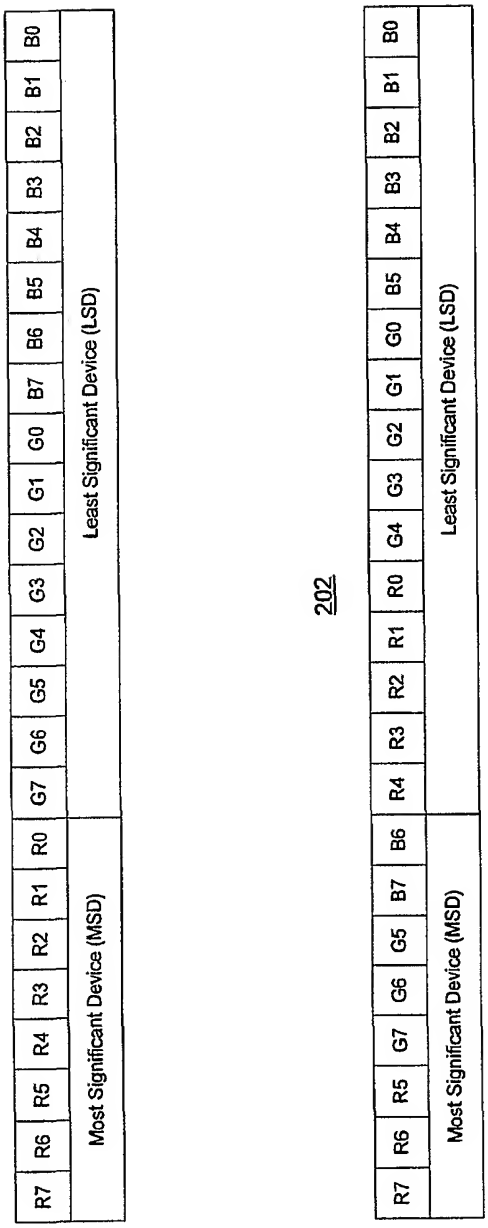
201
R7 R6 R5 R4 R3 R2 R1 R0 G7 G6 G5 G4 G3 G2 G1 G0 B7 B6 B5 B4 B3 B2 B1 B0
Most Significant Device (MSD)
Least Significant Device (LSD)
202
R7 R6 R5 G7 G6 G5 B7 B6 R4 R3 R2 R1 R0 G4 G3 G2 G1 G0 B5 B4 B3 B2 B1 B0
Most Significant Device (MSD)
Least Significant Device (LSD)

Fig. 2

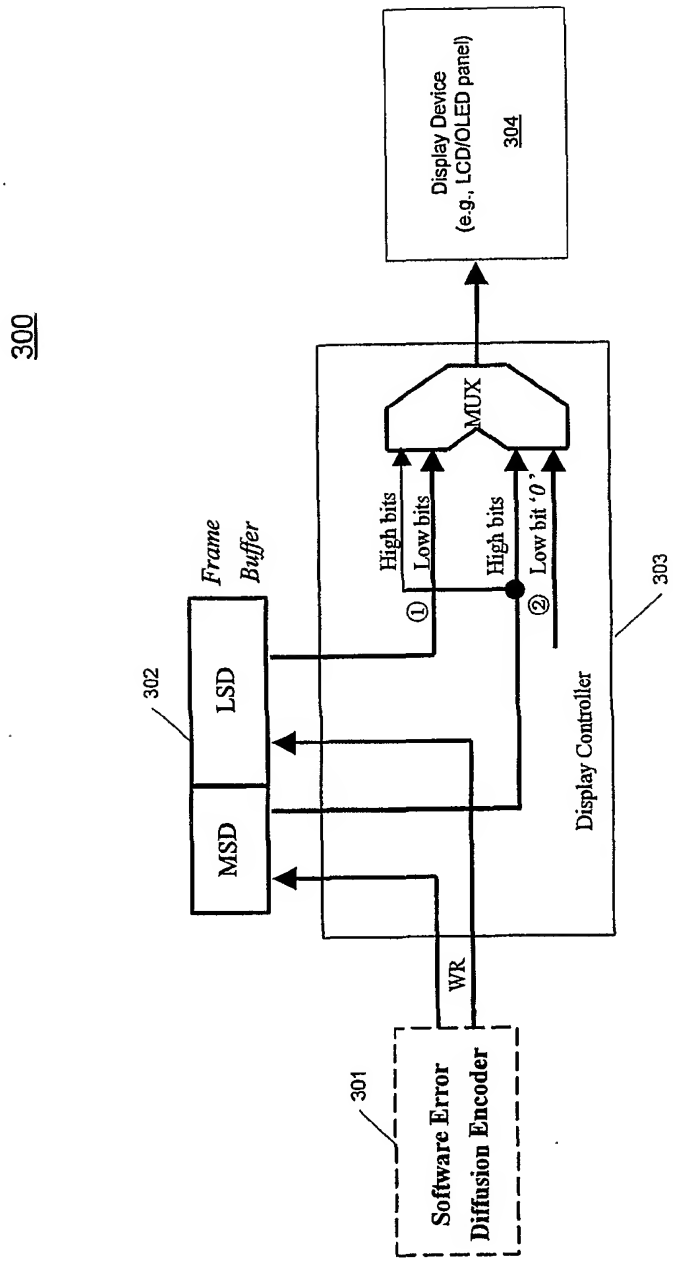
300
Display Device
(e.g., LCD/OLED panel)
304
MUX
High bits
Low bits
High bits
Low bit '0'
①
②
303
Display Controller
Frame
Buffer
302
LSD
MSD
WR
301
Software Error
Diffusion Encoder

Fig. 3

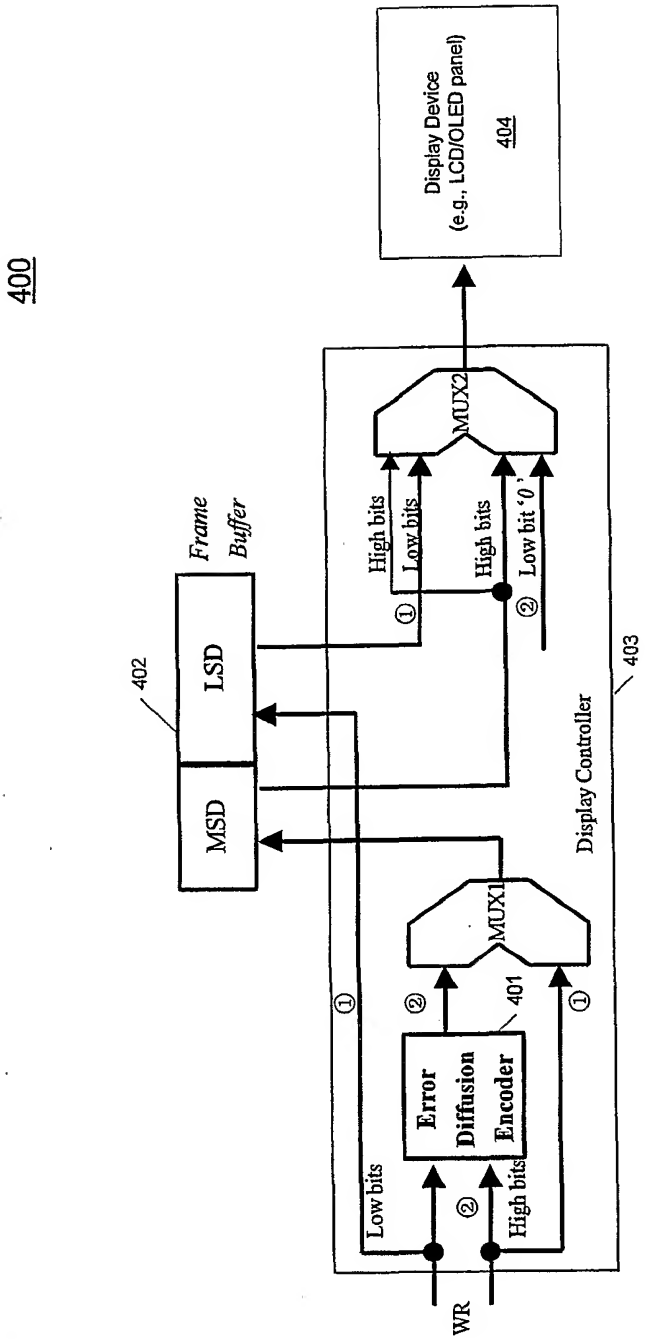
400
Display Device
(e.g., LCD/OLED panel)
404
MUX2
High bits
Low bits
High bits
Low bit '0'
Frame
Buffer
402
LSD
MSD
403
Display Controller
MUX1
401
Error
Diffusion
Encoder
Low bits
High bits
WR

Fig. 4

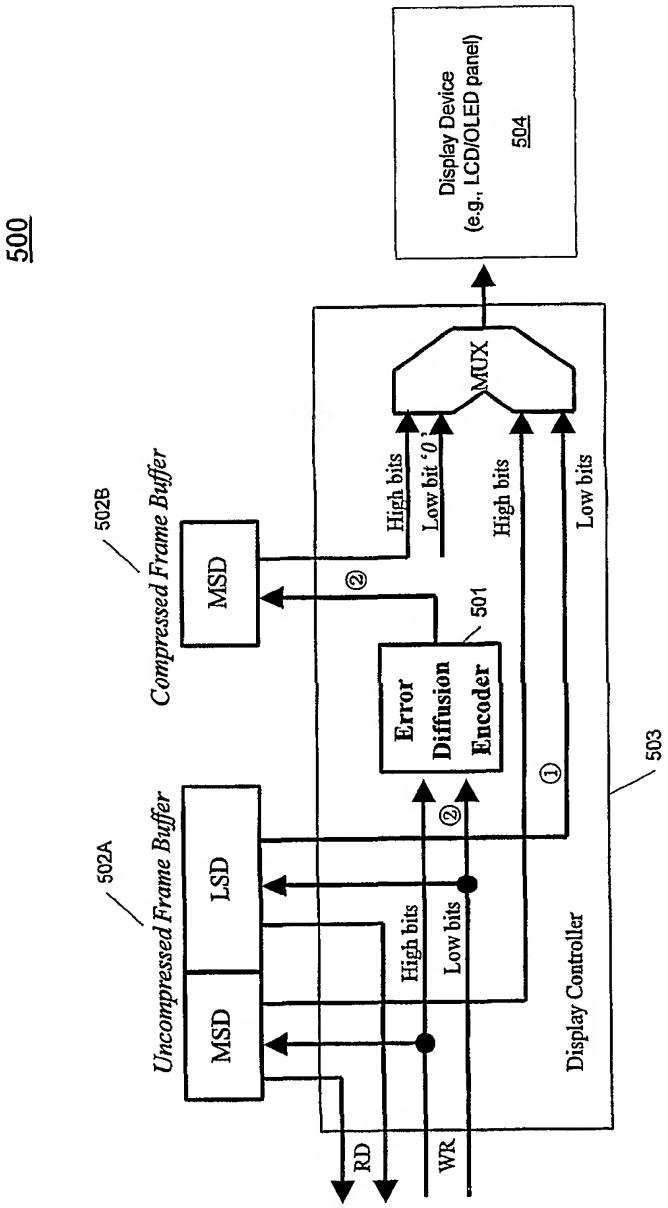
500
Display Device
(e.g., LCD/OLED panel)
504
MUX
High bits
Low bit '0'
High bits
Low bits
502B
Compressed Frame Buffer
MSD
②
501
Error
Diffusion
Encoder
①
503
502A
Uncompressed Frame Buffer
LSD
MSD
②
High bits
Low bits
Display Controller
RD
WR

Fig. 5

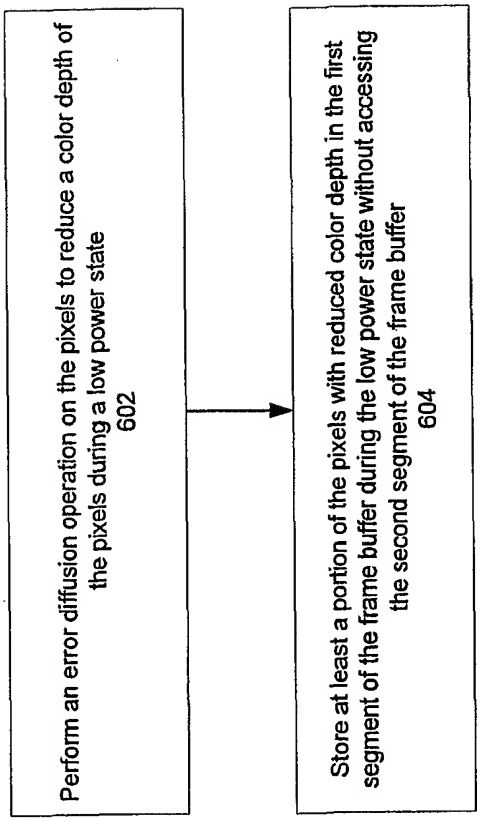
600
Perform an error diffusion operation on the pixels to reduce a color depth of the pixels during a low power state
602
Store at least a portion of the pixels with reduced color depth in the first segment of the frame buffer during the low power state without accessing the second segment of the frame buffer
604

Fig. 6A

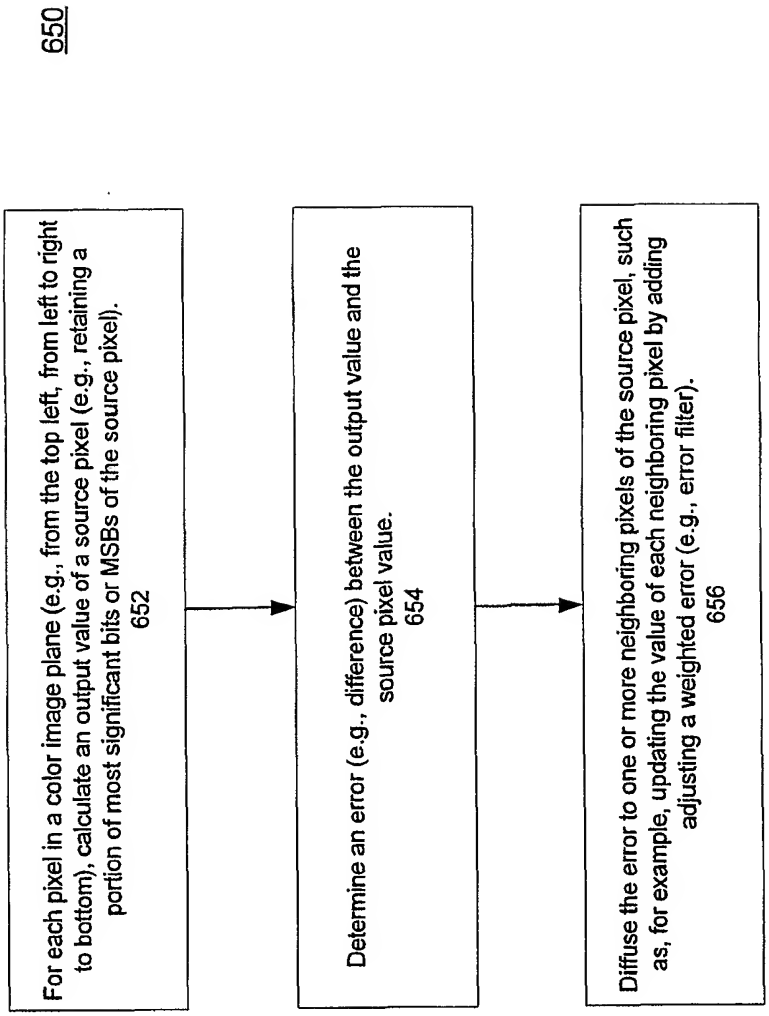
650
For each pixel in a color image plane (e.g., from the top left, from left to right to bottom), calculate an output value of a source pixel (e.g., retaining a portion of most significant bits or MSBs of the source pixel).
652
Determine an error (e.g., difference) between the output value and the source pixel value.
654
Diffuse the error to one or more neighboring pixels of the source pixel, such as, for example, updating the value of each neighboring pixel by adding adjusting a weighted error (e.g., error filter).
656

Fig. 6B

Notation

buffer[]	Memory buffer sized of one image row to store the error diffused to the bottom pixels
register	Register to store the error diffused to the right pixel
FrameBuffer[]	Frame buffer to store the result image pixel, the addressing between scan lines is omitted
SrcImage[]	Original image data, the addressing between scan lines is omitted
temp_sum, *temp-error*	Temporary buffer, can be implemented by register

Algorithm

1. Set all the cells in *buffer*[] to 0
2. For each row in the image
3. Set *register* to 0
4. For each pixel position *x* in one row
5. *temp_sum* = *SrcImage*[x] + *register* + *buffer*[x] (update the current image pixel value)
6. if (overflow in sum operation) *temp_sum* = 255
7. *FrameBuffer*[x] = *temp_sum* & 0xe0 (quantizing to get output pixel value)
8. *temp_error* = *temp_sum* & 0x1f (calculate error)
9. *register* = *temp_error* >> 1 (diffuse error to the right pixel)
10. *buffer*[x] = *temp_error* >> 1 (diffuse error to the bottom pixel)
11. End of For
12. End of For

Fig. 7

Notation

buffer[]	buffer store one line of diffused error for the row below, each cell of the buffer has 2 bits
register	register store diffused error for pixel to the right, 2 bits
FrameBuffer[]	MSB part of the display buffer memory, each cell has 3 bits
SrcImage[]	original image pixel data fed via the display interface to the host system
X[a:b]	b, b+1, ..., a-1, a bit of data X

Algorithm

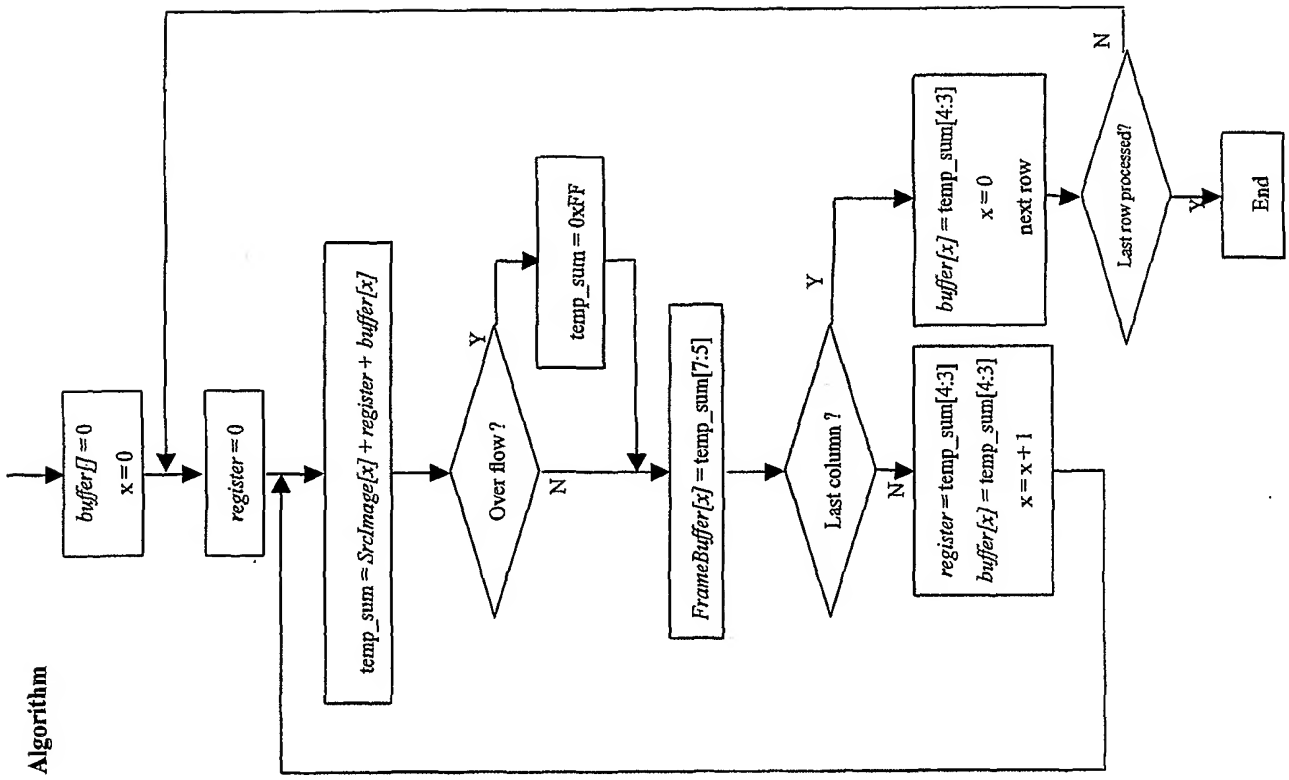

Fig. 8

```
module ErrorDiffuse(R_i, clk, R_o);

input       [7:0]    R_i;
input                clk;
output  [2:0]    R_o;
reg     [1:0]    error1[1:raw],error_n[1];
wire    [7:0]    temp_R;
integer          raw_cnt,line_cnt;

if((line_cnt==1)&&(raw_cnt==1))
  assign temp_R=R_i;
else if((line_cnt==1)&&(raw_cnt!=1))
  assign temp_R=R_i+{error_n[1],2'b00}<R_i?8'hFF:R_i+{error_n[1],2'b00};
else if((line_cnt!=1)&&(raw_cnt==1))
  assign temp_R=R_i+{error1[raw_cnt],2'b00}<R_i?8'hFF:R_i+{error1[raw_cnt],2'b00};
else
  assign temp_R=R_i+{error1[raw_cnt],2'b00}+{error_n[1],2'b00}<R_i?8'hFF:R_i+{error1[raw_cnt],2'b00}+{error_n[1],2'b00};

always @ (posedge clk)
  begin
        error_n[1]    =    temp_R[4:3];
        error1[raw_cnt]    =    error_n[1];
  end
assign R_o = temp_R[7:5];

endmodule
```

Fig. 9

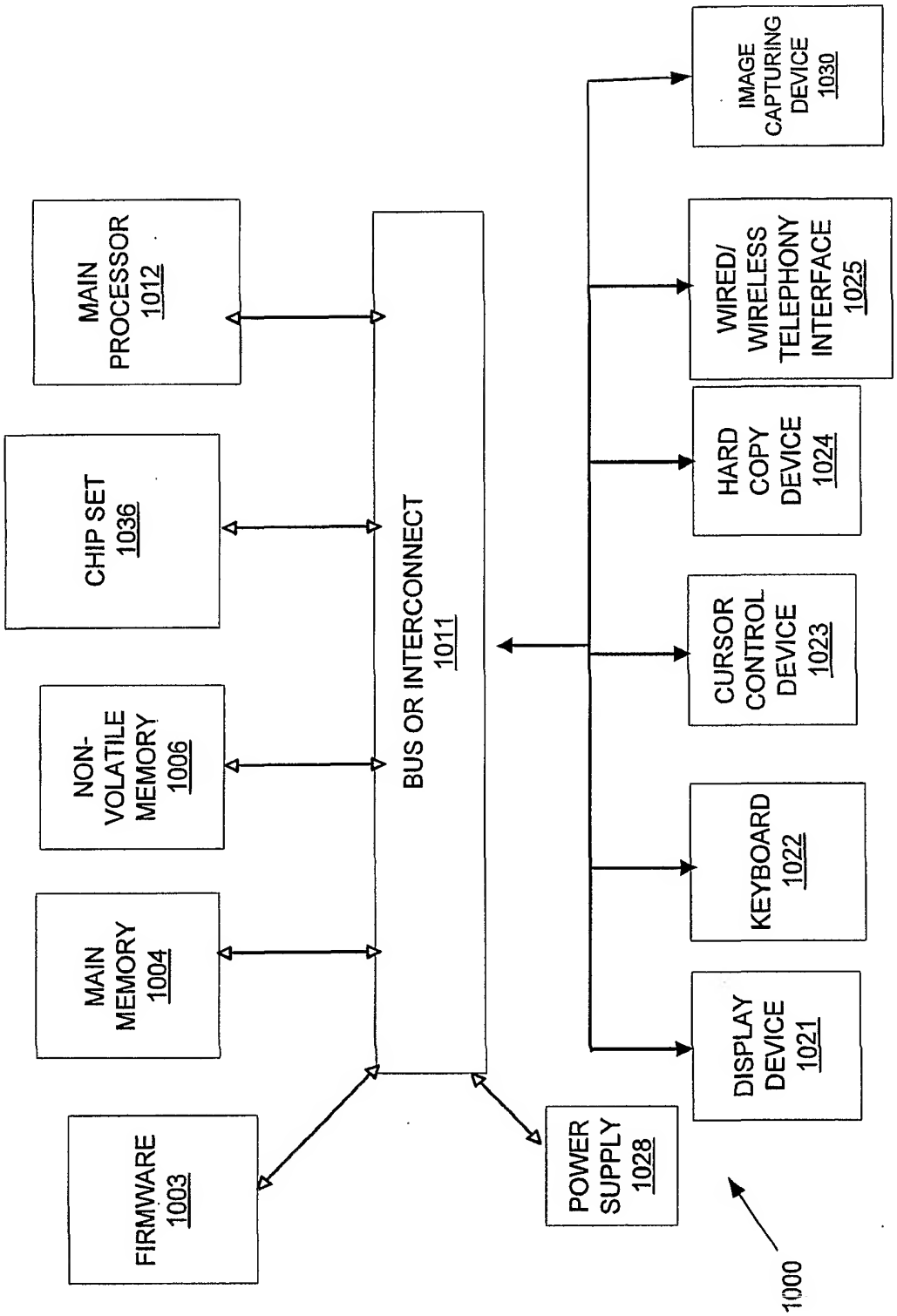
MAIN PROCESSOR 1012
CHIP SET 1036
NON-VOLATILE MEMORY 1006
MAIN MEMORY 1004
FIRMWARE 1003
BUS OR INTERCONNECT 1011
POWER SUPPLY 1028
DISPLAY DEVICE 1021
KEYBOARD 1022
CURSOR CONTROL DEVICE 1023
HARD COPY DEVICE 1024
WIRED/ WIRELESS TELEPHONY INTERFACE 1025
IMAGE CAPTURING DEVICE 1030
1000

FIG. 10